Structure-aware Scene-Graph Conditioning for DiT / SD3

Arnav Shukla (2022103)
Naman Birla (2022310)

Research Papers

SGDiff (Yang et al., 2022): Scene-Graph Diffusion: masked + contrastive SG embeddings for graph to image alignment.

SD3 (Esser et al., 2024): Scaling Rectified Flow Transformers (MM-DiT backbone) for high-quality text to image synthesis. [two stream networks]

T2I Adapter (Mou et al. 2024): T2I Adapter: Learning Adapters to Dig out More Controllable Ability for Text-to-Image Diffusion Models

ControlNet (Zhang et al. 2023): Adding Conditional Control to Text-to-Image Diffusion Models

SD3 MM-DiT (Multimodal diffusion transformer)

- They use a multimodal DiT as a backbone instead of the standard UNet
- Processes both text and image tokens within a shared transformer, enabling bidirectional interaction between modalities.
- Separate linear layers project text and image features into a common embedding space before fusion.
- Joint self-attention over concatenated text and image tokens with modality-specific weights, enabling bidirectional interaction.

$$dy_t = v_\Theta(y_t, t)\, dt \ ,$$

Consider generative models that define a mapping between samples x1 from a noise distribution p1 to samples x0 from a data distribution p0 in terms of an ordinary differential equation (ODE), where the velocity(v) is parameterized by the weights Θ of a neural network.

$$z_t = a_t x_0 + b_t \epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \ .$$

A more efficient alternative is to directly regress a vector field ut that generates a probability path between p0 and p1. To construct such a ut, we define a forward process, corresponding to a probability path pt between p0 and p1 = N(0,1)

For $a_0 = 1, b_0 = 0, a_1 = 0$ and $b_1 = 1$, the marginals,

$$p_t(z_t) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} p_t(z_t|\epsilon) \ , \tag{3}$$

are consistent with the data and noise distribution.

To express the relationship between z_t, x_0 and ϵ, we introduce ψ_t and u_t as

$$\psi_t(\cdot|\epsilon) : x_0 \mapsto a_t x_0 + b_t \epsilon \tag{4}$$

$$u_t(z|\epsilon) \coloneqq \psi_t'(\psi_t^{-1}(z|\epsilon)|\epsilon) \tag{5}$$

$$u_t(z) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0,I)} u_t(z|\epsilon) \frac{p_t(z|\epsilon)}{p_t(z)} \tag{6}$$

While regressing u_t with the *Flow Matching* objective

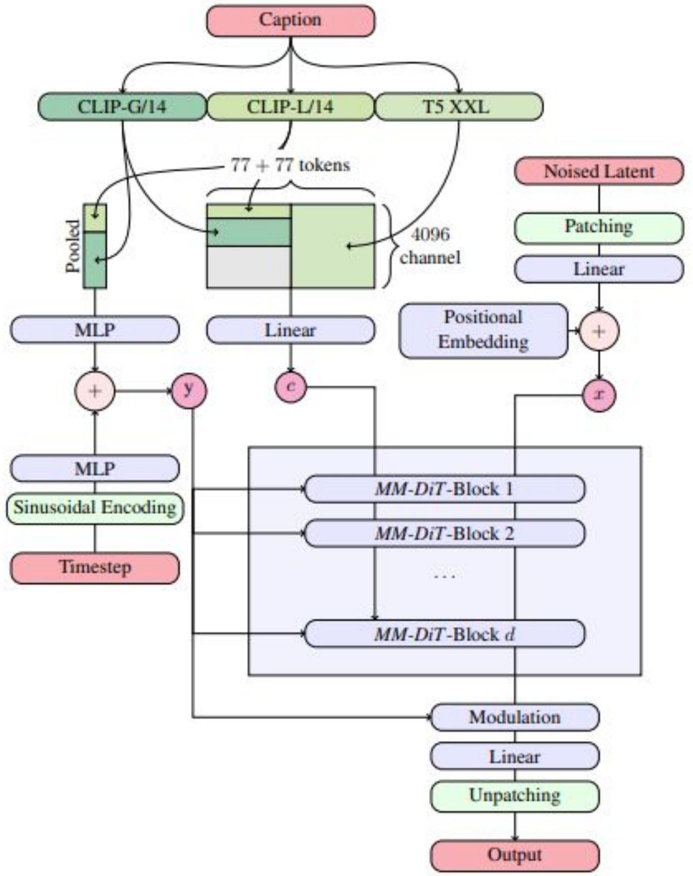
$$\mathcal{L}_{FM} = \mathbb{E}_{t, p_t(z)} ||v_\Theta(z, t) - u_t(z)||_2^2. \tag{7}$$

directly is intractable due to the marginalization in Equation 6, *Conditional Flow Matching* (see B.1),

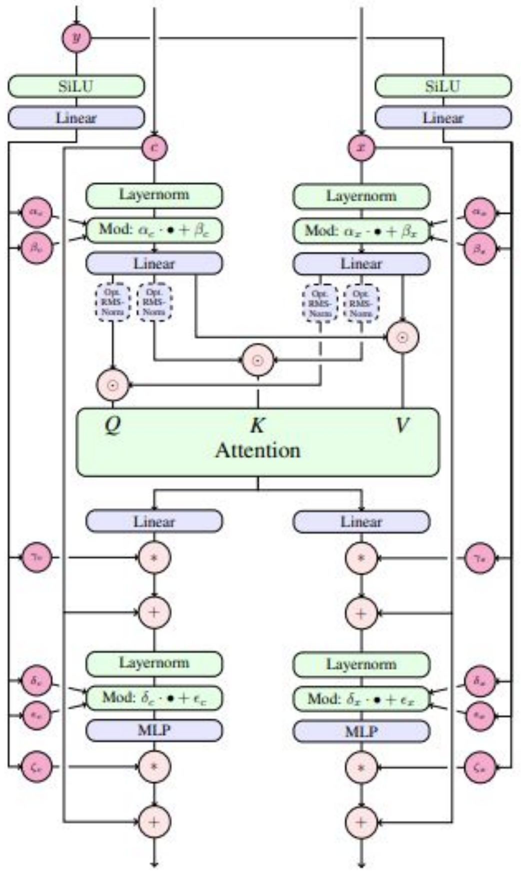
$$\mathcal{L}_{CFM} = \mathbb{E}_{t, p_t(z|\epsilon), p(\epsilon)} ||v_\Theta(z, t) - u_t(z|\epsilon)||_2^2 \ , \tag{8}$$

$$\mathcal{L}_w(x_0) = -\frac{1}{2} \mathbb{E}_{t \sim \mathcal{U}(t), \epsilon \sim \mathcal{N}(0,I)} \left[w_t \lambda_t' \|\epsilon_\Theta(z_t, t) - \epsilon\|^2 \right]$$

where $w_t = -\frac{1}{2} \lambda_t' b_t^2$ corresponds to $\mathcal{L}_{CFM}$.



(a) Overview of all components.


y
SiLU
Linear
c
x
Layernorm
Mod: $\alpha_c \cdot \bullet + \beta_c$
Mod: $\alpha_x \cdot \bullet + \beta_x$
Linear
Opt. RMS-Norm
Q
K
V
Attention
Layernorm
Mod: $\delta_c \cdot \bullet + \epsilon_c$
Mod: $\delta_x \cdot \bullet + \epsilon_x$
MLP

(b) One *MM-DiT* block

Figure 2. **Our model architecture.** Concatenation is indicated by $\odot$ and element-wise multiplication by $*$. The RMS-Norm for Q and K can be added to stabilize training runs. Best viewed zoomed in.

SGDiff- Diffusion-Based Scene Graph to Image Generation

- Idea is to learn scene-graph embeddings (local + global) via masked autoencoding and contrastive pre-training, then condition a latent diffusion model on those embeddings to generate images.
- Two-stage pipeline: (1) SG encoder (GCN) pre-trained with masked contrastive objective → produces triplet/local and global embeddings; (2) latent diffusion model (VAE + U-Net) conditioned on SG embeddings for image synthesis.
- editing nodes/relations in the scene graph yields predictable edits in generated images - supports controllable, compositional synthesis.
- **Embeddings can be used as conditioning for other models**

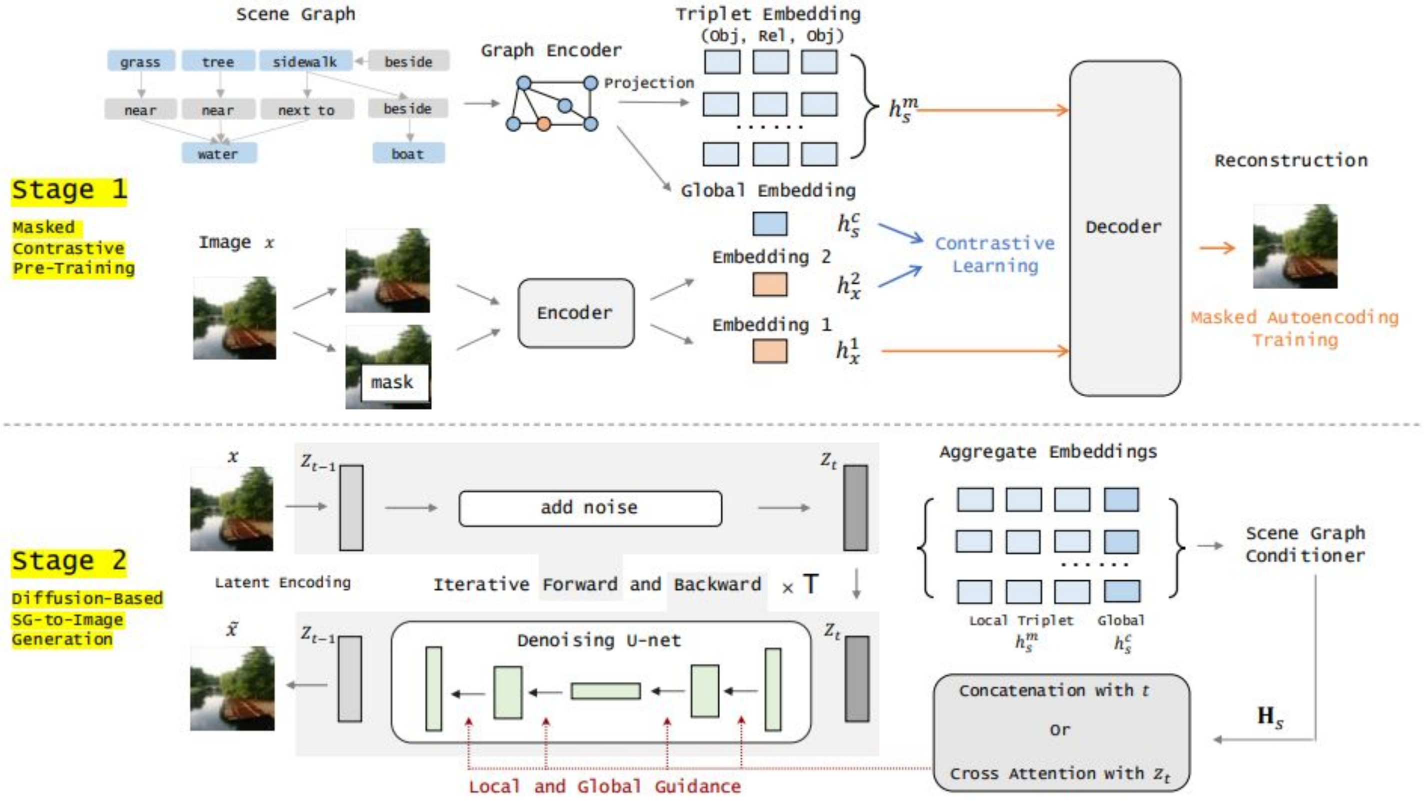
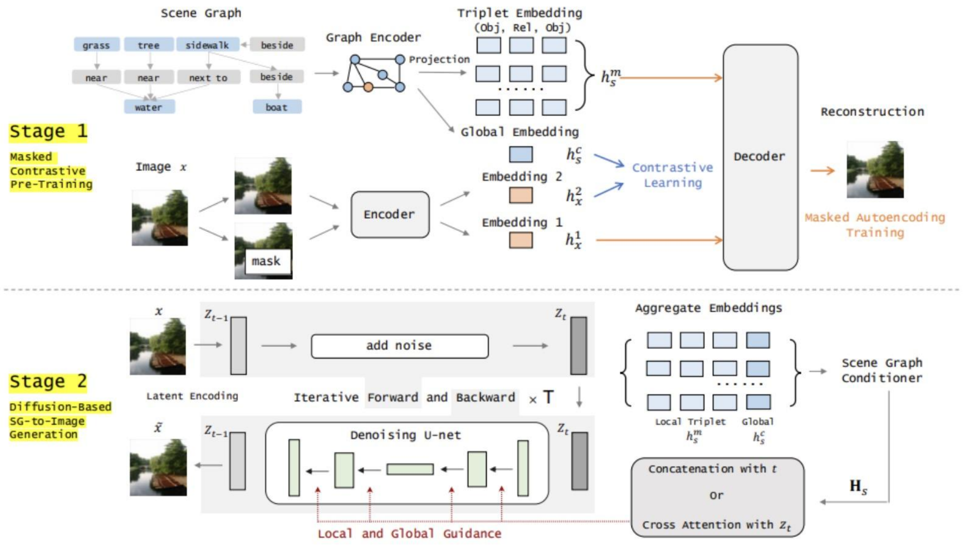


Figure 2. **Schematic illustration of SGDiff.** In the first stage, we pre-train a scene graph encoder using the combination of a masked autoencoding loss and a contrastive loss. In the second stage, we build a latent diffusion model that conditions on embeddings produced by the scene graph encoder.

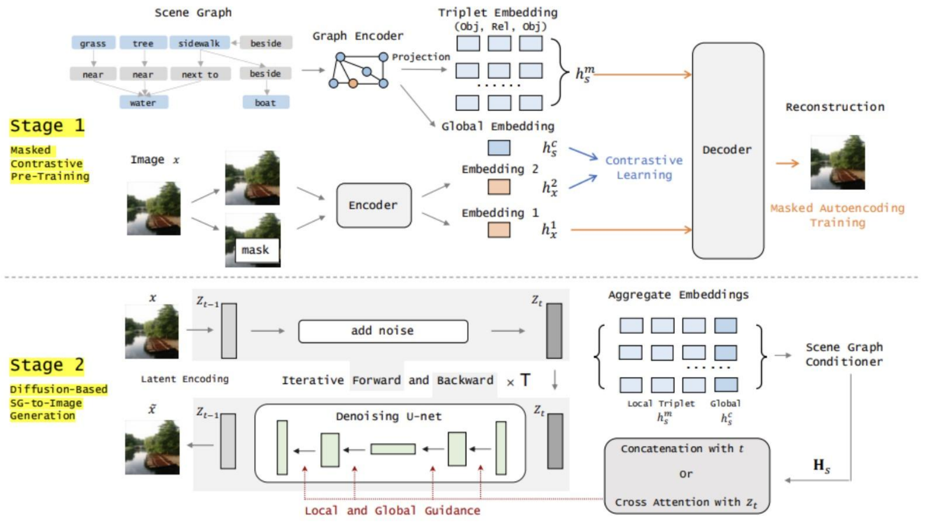
$$h_{o_i} = \text{Pool}(\{f_o^{out}(h_{o_i}, h_{r_{ij}}, h_{o_j})\}_{j\in\mathcal{N}_{out}(o_i)} \cup f_o^{in}(h_{o_j}, h_{r_{ji}}, h_{o_i})\}_{j\in\mathcal{N}_{in}(o_i)}),$$

$$h_{r_{ij}} = f_r(h_{o_j}, h_{r_{ij}}, h_{o_i}),$$



Pre-training (masked):

1. In particular, we randomly choose a triplet (oi , rij , oj) in the scene graph s
2. Mask out objects oi and oj in the corresponding scene image x.
3. We denote this masked area as xm, and the remaining image as x\m.
4. To train the SG encoder, we consider the task of predicting xm from x\m and embeddings obtained from the SG encoder => trained on L2 loss [**LOCAL**]



Pre-training (contrastive):

compute a graph-level embedding by concatenating all object and relation embeddings obtained from the SG encoder

Choose uniformly negative samples from dataset. We already have the positive samples.

A negative sample consists of an object/relation not present in the image. Train this using contrastive loss. [**GLOBAL**]

$$h_s^c = f^c(\texttt{concat}(\mathbf{Pool}(\{h_o\}_{o\in\mathcal{O}}),\ \mathbf{Pool}(\{h_r\}_{r\in\mathcal{R}}))),$$

$$\mathcal{L}_{\text{masked}} = \mathop{\mathbb{E}}_{(s,x)\sim D} \|x_m - d_\theta(x_{\setminus m}, h_s^m)\|_2^2,$$

$$\mathcal{L}_{\text{contrastive}}(f;\tau,k) = \mathop{\mathbb{E}}_{\substack{(s,x^+)\sim D \\ \{x_i^-\}_{i=1}^k \sim D_s}} \left[-\log \frac{\exp({h_s^c}^\top h_{x^+}/\tau)}{\exp\left({h_s^c}^\top h_{x^+}/\tau\right) + \sum_i \exp\left({h_s^c}^\top h_{x_i^-}/\tau\right)} \right],$$

$$\boxed{\mathcal{L} = \mathcal{L}_{\text{masked}} + \lambda \mathcal{L}_{\text{contrastive}},}$$

T2I Adapter - Adapter for Text-to-Image Diffusion Model

- Goal is to enable structure guided controllable image generation without retraining the whole network,
- Adds trainable lightweight adapter networks and keeps the main diffusion backbone as frozen
- Each of the adapter represents a control type and the features from these networks are embedded into the Unet feature maps [concat or cross attention]
- **Inspires LORA based embedding injection**

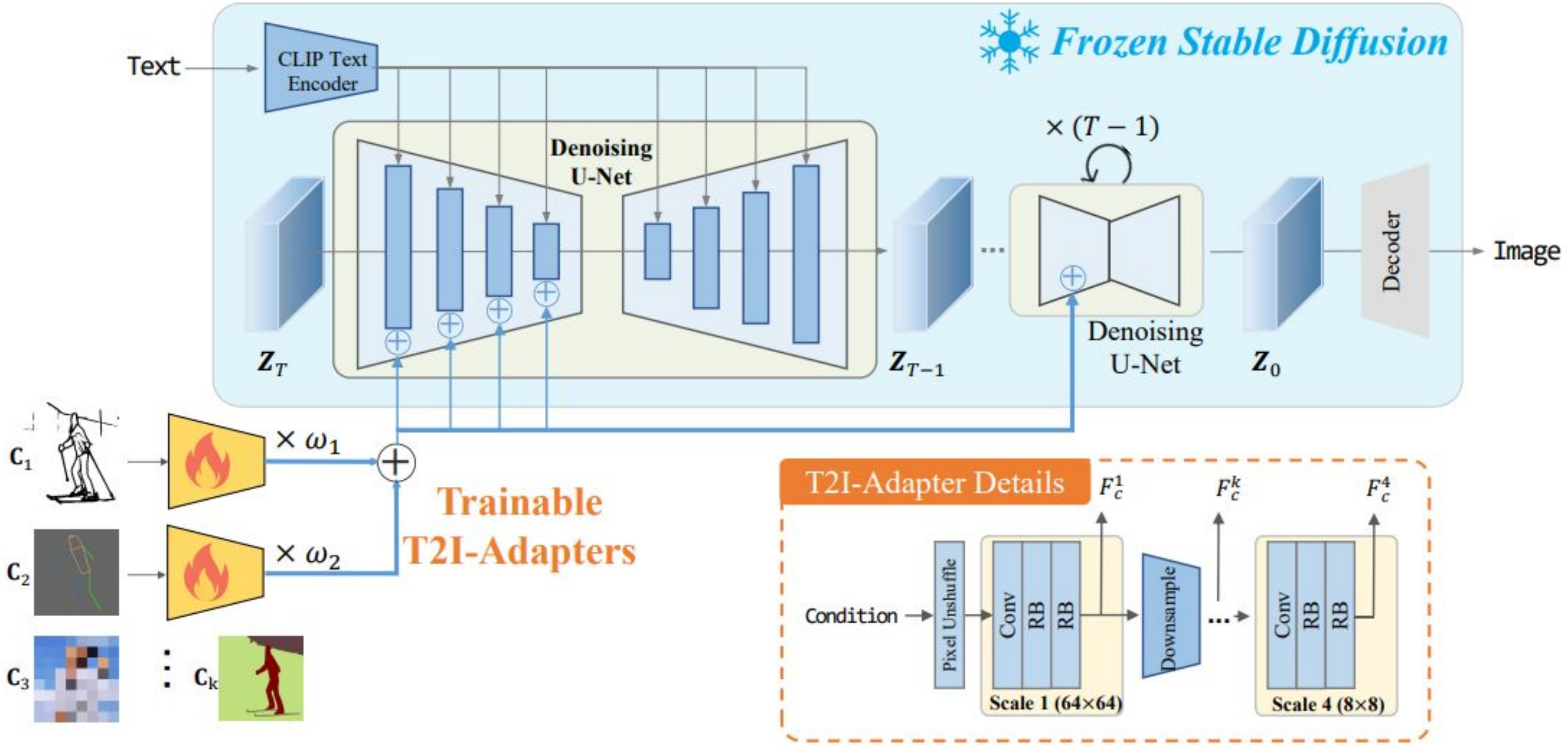


Figure 3. The overall architecture is composed of two parts: 1) a pre-trained stable diffusion model with fixed parameters; 2) several T2I-Adapters trained to align internal knowledge in T2I models and external control signals. Different adapters can be composed by directly adding with adjustable weight ω. The detailed architecture of T2I-Adapter is shown in the lower right corner.

ControlNet - Adding Conditional Control to Text-to-Image Diffusion Models

- Provide precise, low-level control (edges, depth, segmentation, poses) for pre-trained diffusion models while preserving their generative knowledge.
- Attach a trainable control network (a copy/side-branch of the UNet) that processes the conditioning map and injects lateral feature maps into the frozen/fine-tuned main UNet.
- Zero-init: ControlNet uses zero-initialized weights (and special “zero conv” layers) so the pretrained diffusion model’s behavior is unchanged at initialization; training the control branch gradually adds conditioning influence without destabilizing generation.

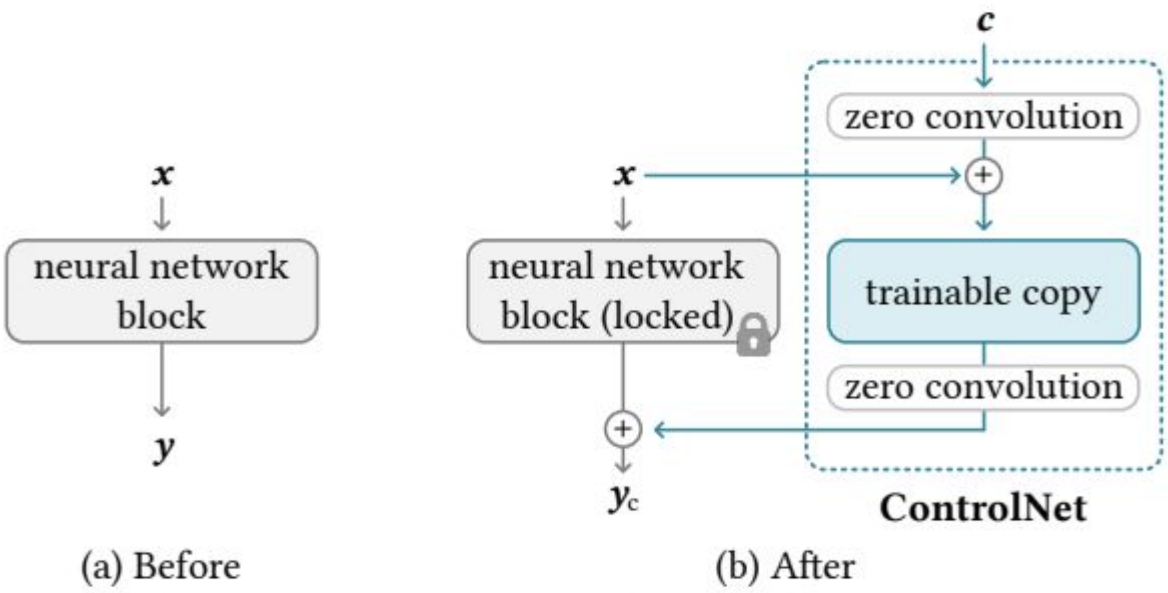
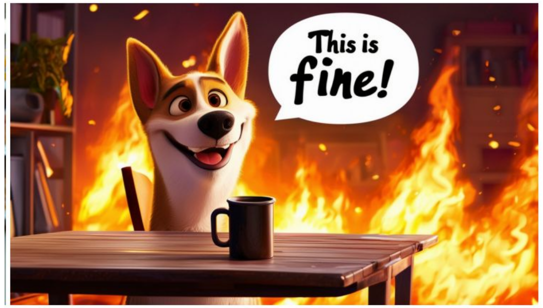


Figure 2: A neural block takes a feature map x as input and outputs another feature map y, as shown in (a). To add a ControlNet to such a block we lock the original block and create a trainable copy and connect them together using zero convolution layers, *i.e.*, 1×1 convolution with both weight and bias initialized to zero. Here c is a conditioning vector that we wish to add to the network, as shown in (b).

Problem Introduction

Text-to-image Diffusion Transformers (e.g., SD3’s MMDiT) generate high-quality images but often miss fine-grained inter-object relationships under prompt-only control. Prior work on scene-graph conditioning (SGDiff) shows that graph encoders can stabilize structural compliance. Other works like T2I Adapters shows that injecting learnable embeddings to a frozen diffusion model improves control over image generation with respect to structures and relations between objects.



smiling cartoon dog sits at a table, coffee mug on hand, as a room goes up in flames. “This is fine,” the dog assures himself.

smiling cartoon dog sits at a table, coffee mug on hand, as a room goes up in flames. “This is fine,” the dog assures himself. [we see that mug-on-hand is not respected in the output].
Ref: https://arxiv.org/pdf/2403.03206 [SD3]

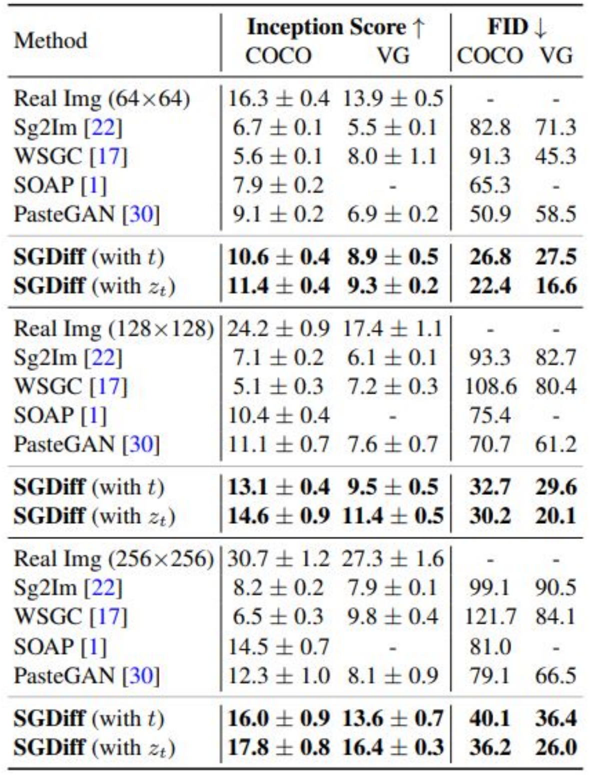
Details of Datasets

- Visual Genome [https://homes.cs.washington.edu/~ranjay/visualgenome/index.html]
    - 108K Image-scene graph pairs with dense annotations: object labels, attributes, relations, bounding boxes - typically used for layout/scene understanding models
- COCO Stuff Dataset [https://github.com/nightrome/cocostuff]
    - COCO dataset with additional data
    - 164 K images along with complex spatial context between stuff and things, used for scene understanding

Results from paper

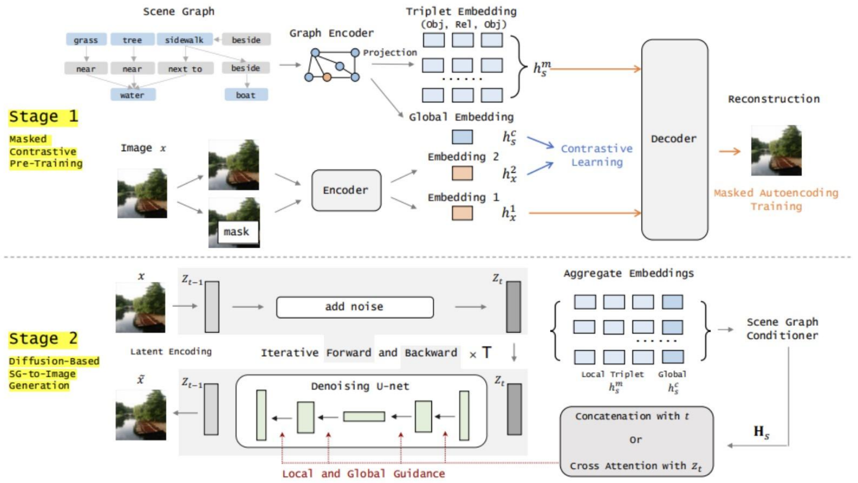
SGDiff significantly improves Inception Score (↑) and lowers FID (↓) compared to Sg2Im, PasteGAN, WSGC, etc.

SGDiff beats prior methods at smaller sizes too



Method	Inception Score ↑ COCO	Inception Score ↑ VG	FID ↓ COCO	FID ↓ VG
Real Img (64×64)	16.3 ± 0.4	13.9 ± 0.5	-	-
Sg2Im [22]	6.7 ± 0.1	5.5 ± 0.1	82.8	71.3
WSGC [17]	5.6 ± 0.1	8.0 ± 1.1	91.3	45.3
SOAP [1]	7.9 ± 0.2	-	65.3	-
PasteGAN [30]	9.1 ± 0.2	6.9 ± 0.2	50.9	58.5
SGDiff (with t)	$\mathbf{10.6 \pm 0.4}$	$\mathbf{8.9 \pm 0.5}$	**26.8**	**27.5**
SGDiff (with z_t)	$\mathbf{11.4 \pm 0.4}$	$\mathbf{9.3 \pm 0.2}$	**22.4**	**16.6**
Real Img (128×128)	24.2 ± 0.9	17.4 ± 1.1	-	-
Sg2Im [22]	7.1 ± 0.2	6.1 ± 0.1	93.3	82.7
WSGC [17]	5.1 ± 0.3	7.2 ± 0.3	108.6	80.4
SOAP [1]	10.4 ± 0.4	-	75.4	-
PasteGAN [30]	11.1 ± 0.7	7.6 ± 0.7	70.7	61.2
SGDiff (with t)	$\mathbf{13.1 \pm 0.4}$	$\mathbf{9.5 \pm 0.5}$	**32.7**	**29.6**
SGDiff (with z_t)	$\mathbf{14.6 \pm 0.9}$	$\mathbf{11.4 \pm 0.5}$	**30.2**	**20.1**
Real Img (256×256)	30.7 ± 1.2	27.3 ± 1.6	-	-
Sg2Im [22]	8.2 ± 0.2	7.9 ± 0.1	99.1	90.5
WSGC [17]	6.5 ± 0.3	9.8 ± 0.4	121.7	84.1
SOAP [1]	14.5 ± 0.7	-	81.0	-
PasteGAN [30]	12.3 ± 1.0	8.1 ± 0.9	79.1	66.5
SGDiff (with t)	$\mathbf{16.0 \pm 0.9}$	$\mathbf{13.6 \pm 0.7}$	**40.1**	**36.4**
SGDiff (with z_t)	$\mathbf{17.8 \pm 0.8}$	$\mathbf{16.4 \pm 0.3}$	**36.2**	**26.0**

Table 1. **Inception Scores and FIDs on COCO-Stuff and VG datasets.** Results of previous methods are either directly taken from their original papers or obtained by running official open-source implementations.

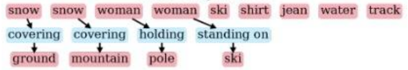


Objective
Generate images from scene graphs using the pretrained SGDiff model on Visual Genome dataset.

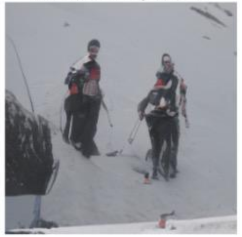
Scene graph encoder
VAE autoencoder: Vector-Quantized fp8 VAE
Sampling method: DDIM with 200 steps
Output resolution: 256×256 pixels
Color space: RGB
Latent space: 4×32×32 (compressed representation)
Working latent shape: (1, 4, 32, 32) = 4096 dimensions

Scene Graph Structure
Objects: Up to 30 objects per scene (e.g., person, car, tree, building, sky)
Object categories: 179 object types (from vocabulary)
Relationships: Spatial and semantic relationships between objects
  - Examples: "person riding bike", "tree near building", "car on road"
Relationship types: 46 predicate types (from vocabulary)
Bounding boxes: Spatial location information for each object

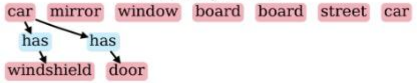
Scene Graph #2500



Generated Image #2500

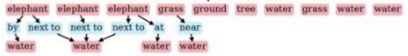


Scene Graph #2510



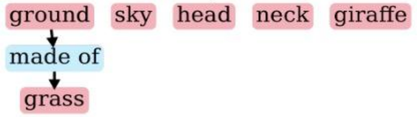
Generated Image #2510

Scene Graph #2505



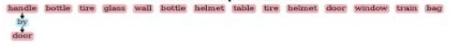
Generated Image #2505

Scene Graph #2520



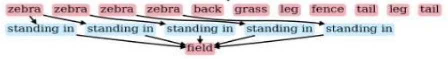
Generated Image #2520

Scene Graph #2530



Generated Image #2530

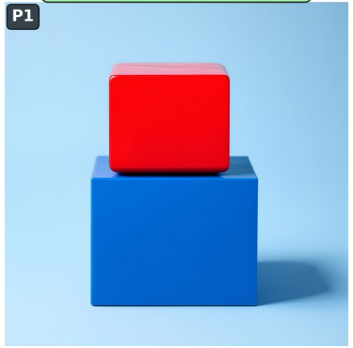
Scene Graph #2540



Generated Image #2540

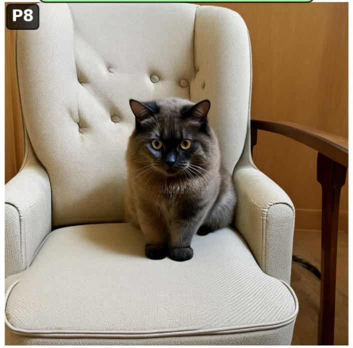
SD3 MM-DiT: Text-to-Image Generation Examples
Structural Understanding Analysis

Category: Spatial Relation



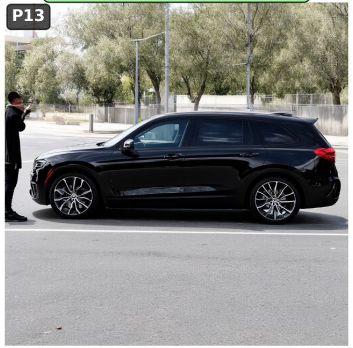
"a red cube on top of a blue cube, plain background"

Category: Contact/Support



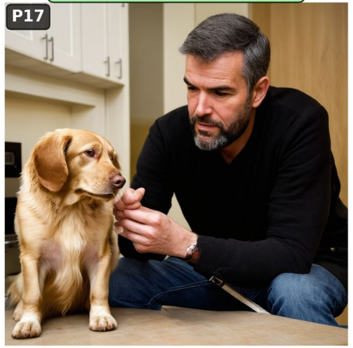
"a cat sitting on a chair"

Category: Natural Objects



"a person standing to the left of a car"

Category: Gaze/Direction



"a man looking at a dog"

Model: stabilityai/stable-diffusion-3-medium-diffusers | Resolution: 1024×1024 | Steps: 28 | Guidance: 7.0 | Seed: 0

Observation

Structural Variation Analysis

1. Images with different object compositions show distinct visual content
2. Number of objects correlates with image complexity

Visual Correspondence

1. Predicates like "near", "on", "above" influence object placement
2. Overall graph structure determines image composition

Consistency Test

1. Each unique scene graph generates a unique image
2. The same scene graph (if rerun) produces similar images with stochastic variations

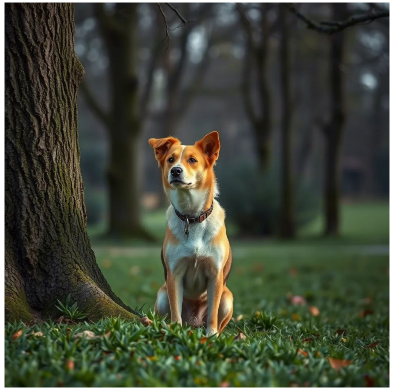
Next Steps

1. To analyze SD3 generation process and get attention maps of different block to understand which layer contribute the most to the structural generation part (ie. interaction of objects and consider simpler relationships only like looking at, touching etc.)
2. Once we identify these blocks, train LoRA weights for one sample, with its graph embedding fed to the DiT.
3. Analyse the change in output with changing graphs and keeping the input prompt the same.
4. Relationship aware feature injection can also be done (ref: DreamRelation: Relation-Centric Video Customization, -> it's a video model, similar image adaptation can be done.

Analysing Attention in MMDiT based Models

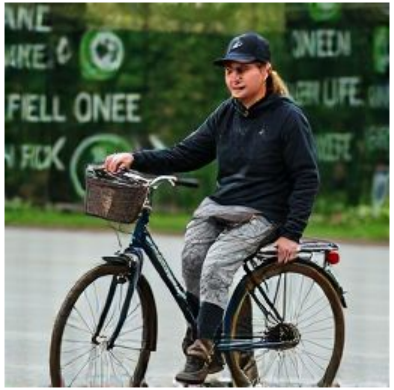
Model: Flux-1 Schnell - 12B parameters
[https://huggingface.co/black-forest-labs/FLUX.1-schnell]

- 19 Double Stream MMDiT blocks [the focus of this project]
- 38 Single Stream Blocks
- Can generate high quality images in 0-4 inference steps



Prompt: A dog standing by a tree [left]

Prompt: A person standing next to a bike [right] **[does not respect structural relationships]**

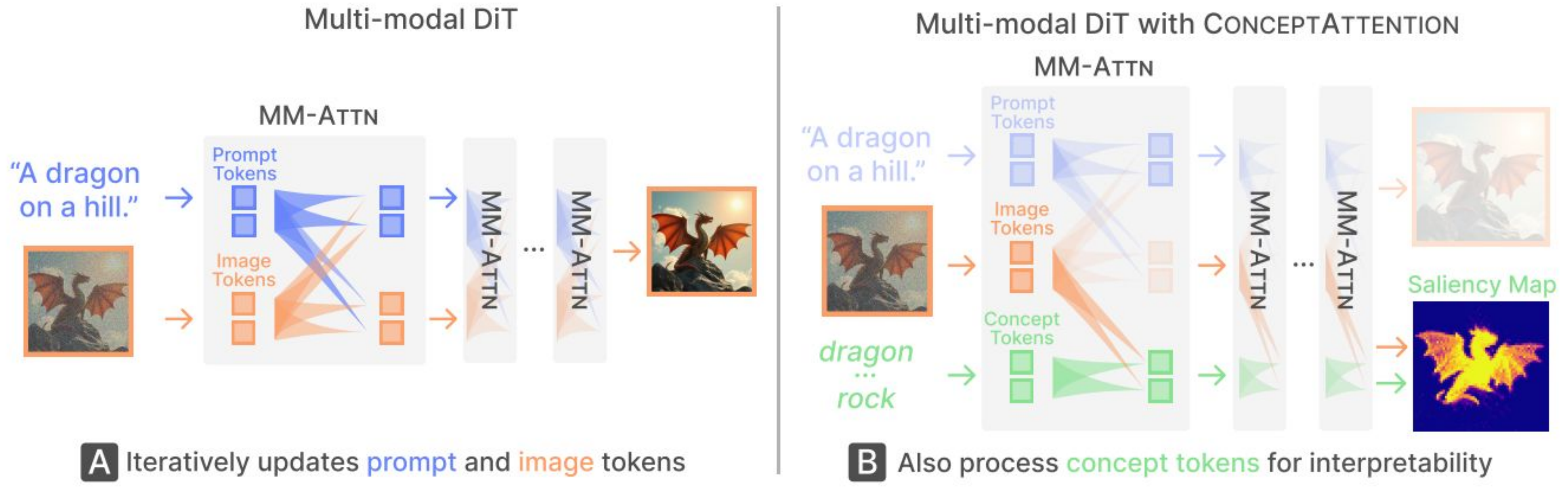


ConceptAttention: Diffusion Transformers Learn Highly Interpretable Features

- DiT models (Flux, SD3) are powerful but we need interpretable maps linking prompt tokens to image regions.
- The better we are able to interpret which layers of the model contribute more to specific parts of image generation, the better we can exploit the model's architecture.
- Key idea is to create contextualized concept tokens and project image patch outputs into the attention output space to compute saliency maps

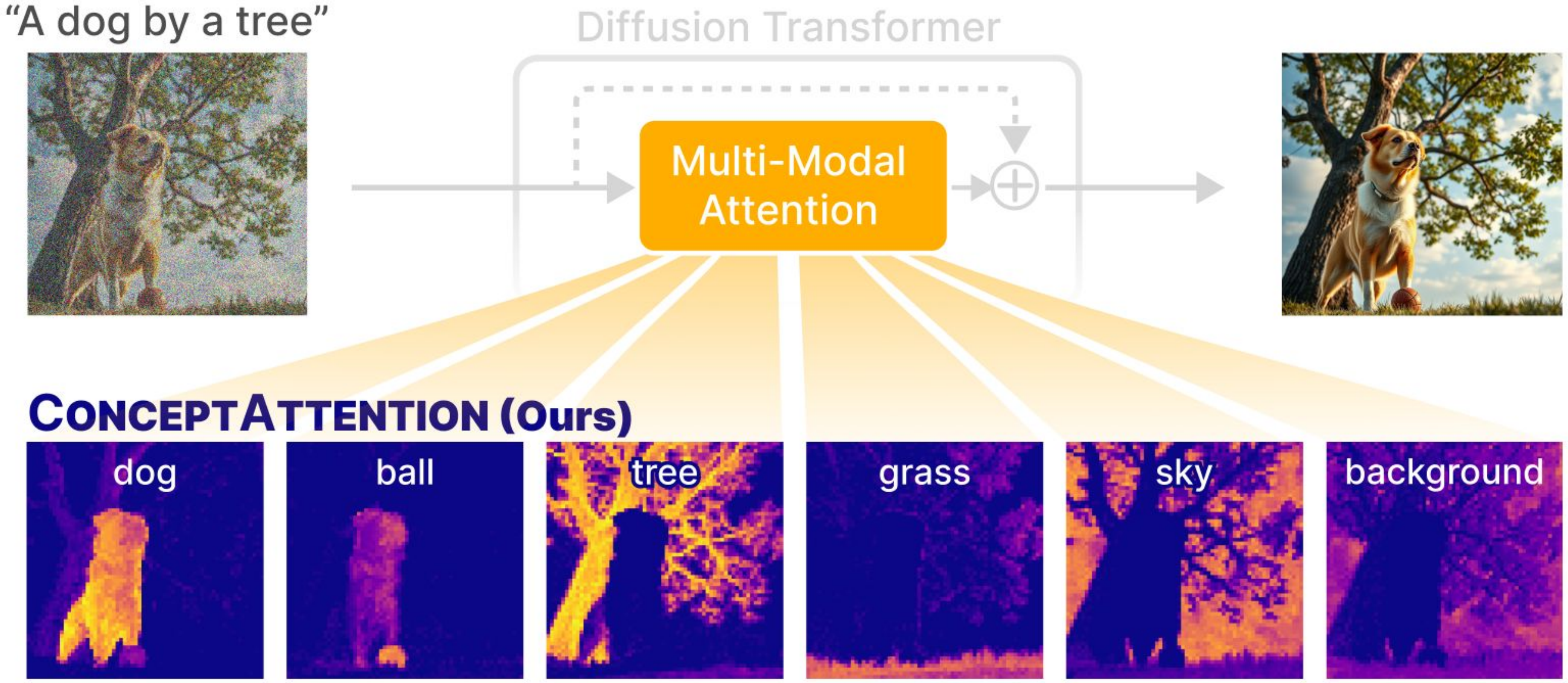
Reference: https://arxiv.org/abs/2502.04320

How it works? [1/2]



CONCEPTATTENTION augments multi-modal DiTs with a sequence of concept embeddings that can be used to produce saliency maps. (Left) An unmodified multi-modal attention (MMATTN) layer processes both prompt and image tokens. (Right) CONCEPTATTENTION augments these layers without impacting the image appearance to create a set of contextualized concept tokens.

ConceptAttention interprets the representations of diffusion transformers by producing saliency maps of text concepts.



How it works? [2/2]

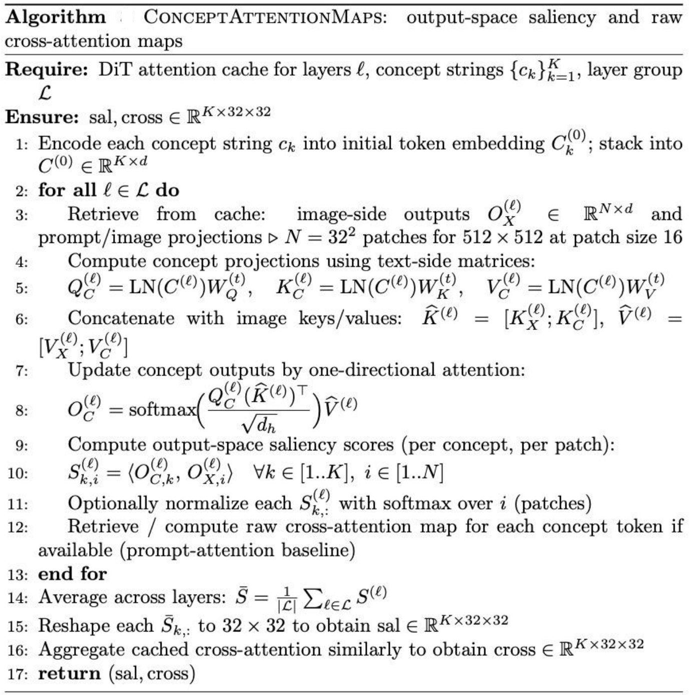
- We supply single-token concepts (eg. dog, ball etc) which are encoded via T5 encoder.

- For each DiT MMAttn layer: use the text projection matrices **W_q^text**, **W_k^text, W_v^text** (reused) to compute Q/K/V for concept tokens. These concept tokens are processed in parallel with prompt and image tokens but constrained so they do not affect image residual streams.

- Collect the attention output vectors (post attention and linear output projection). Compute a linear projection (dot product / projection) between these output vectors and image patch representations to score the presence of each concept per patch [**saliency maps**]

METHODOLOGY

- We want to analyze which layers contribute most to the structural generation part.
- We use the saliency maps to try to localize which "part" of the model has the strongest connection to structural relationships as well as the layers having weaker connection to the structural relationship
- **We split the models 19 MMDIT layers/blocks into 3 parts:**
    - **Early layers -> Blocks [0,1,2,3,4,5,6]**
    - **Middle layers -> Blocks [7,8,9,10,11,12]**
    - **Late layers -> Blocks [13,14,15,16,17,18]**



Algorithm ConceptAttentionMaps: output-space saliency and raw cross-attention maps

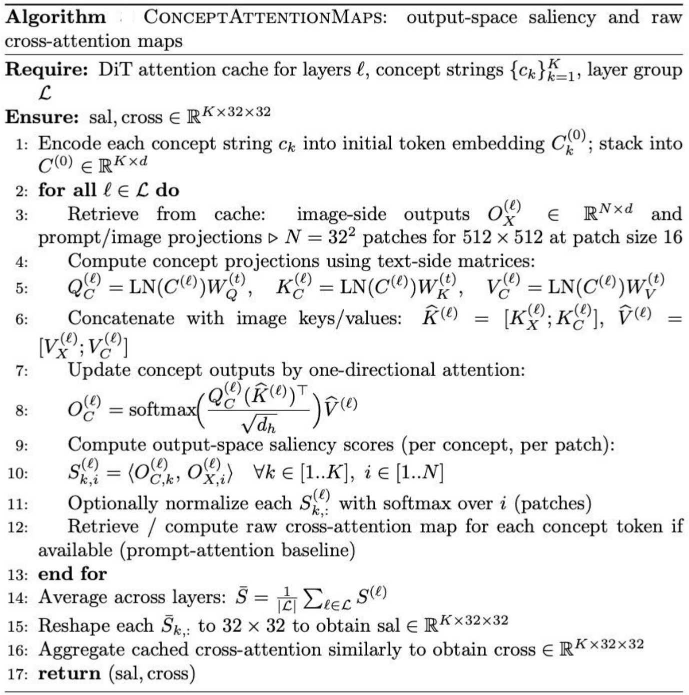
Require: DiT attention cache for layers ℓ, concept strings $\{c_k\}_{k=1}^{K}$, layer group $\mathcal{L}$

Ensure: $\mathrm{sal}, \mathrm{cross} \in \mathbb{R}^{K\times 32\times 32}$

```
1: Encode each concept string $c_k$ into initial token embedding $C_k^{(0)}$; stack into $C^{(0)} \in \mathbb{R}^{K\times d}$
2: for all $\ell \in \mathcal{L}$ do
3:     Retrieve from cache: image-side outputs $O_X^{(\ell)} \in \mathbb{R}^{N\times d}$ and prompt/image projections $\triangleright$ $N = 32^2$ patches for $512\times 512$ at patch size 16
4:     Compute concept projections using text-side matrices:
5:     $Q_C^{(\ell)} = \mathrm{LN}(C^{(\ell)})W_Q^{(t)}, \quad K_C^{(\ell)} = \mathrm{LN}(C^{(\ell)})W_K^{(t)}, \quad V_C^{(\ell)} = \mathrm{LN}(C^{(\ell)})W_V^{(t)}$
6:     Concatenate with image keys/values: $\widehat{K}^{(\ell)} = [K_X^{(\ell)}; K_C^{(\ell)}]$, $\widehat{V}^{(\ell)} = [V_X^{(\ell)}; V_C^{(\ell)}]$
7:     Update concept outputs by one-directional attention:
8:     $O_C^{(\ell)} = \mathrm{softmax}\Big(\frac{Q_C^{(\ell)}(\widehat{K}^{(\ell)})^\top}{\sqrt{d_h}}\Big)\widehat{V}^{(\ell)}$
9:     Compute output-space saliency scores (per concept, per patch):
10:    $S_{k,i}^{(\ell)} = \langle O_{C,k}^{(\ell)}, O_{X,i}^{(\ell)}\rangle \quad \forall k \in [1..K],\ i \in [1..N]$
11:    Optionally normalize each $S_{k,:}^{(\ell)}$ with softmax over $i$ (patches)
12:    Retrieve / compute raw cross-attention map for each concept token if available (prompt-attention baseline)
13: end for
14: Average across layers: $\bar{S} = \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}} S^{(\ell)}$
15: Reshape each $\bar{S}_{k,:}$ to $32\times 32$ to obtain $\mathrm{sal} \in \mathbb{R}^{K\times 32\times 32}$
16: Aggregate cached cross-attention similarly to obtain $\mathrm{cross} \in \mathbb{R}^{K\times 32\times 32}$
17: return $(\mathrm{sal}, \mathrm{cross})$
```

METHODOLOGY

- We curate our **own dataset**
- Using Visual Genome dataset, we take the objects and relationships to create saliency maps of samples from **24 relationship classes.**
- **700 saliency maps & cross attention maps for each class and this was repeated for all 3 batches of layers [total: 700 *24 * 3 maps]**
- Sample prompt: a photo of mug above desk on a plain background
- “A photo of <object_i> <relationship r_ij> <object_j> on a plain background” with concepts: [<object_i>, <relationship r_ij>, <object_j>]



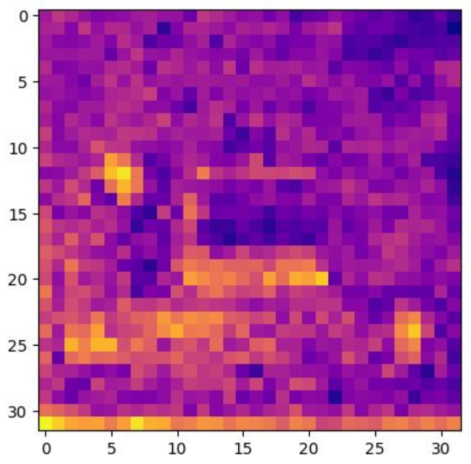
```
Algorithm  ConceptAttentionMaps: output-space saliency and raw cross-attention maps
Require: DiT attention cache for layers $\ell$, concept strings $\{c_k\}_{k=1}^{K}$, layer group $\mathcal{L}$
Ensure: $\mathrm{sal}, \mathrm{cross} \in \mathbb{R}^{K\times 32\times 32}$
 1: Encode each concept string $c_k$ into initial token embedding $C_k^{(0)}$; stack into $C^{(0)} \in \mathbb{R}^{K\times d}$
 2: for all $\ell \in \mathcal{L}$ do
 3:     Retrieve from cache: image-side outputs $O_X^{(\ell)} \in \mathbb{R}^{N\times d}$ and prompt/image projections $\triangleright$ $N = 32^2$ patches for $512\times 512$ at patch size 16
 4:     Compute concept projections using text-side matrices:
 5:     $Q_C^{(\ell)} = \mathrm{LN}(C^{(\ell)})W_Q^{(t)}, \quad K_C^{(\ell)} = \mathrm{LN}(C^{(\ell)})W_K^{(t)}, \quad V_C^{(\ell)} = \mathrm{LN}(C^{(\ell)})W_V^{(t)}$
 6:     Concatenate with image keys/values: $\widehat{K}^{(\ell)} = [K_X^{(\ell)}; K_C^{(\ell)}]$, $\widehat{V}^{(\ell)} = [V_X^{(\ell)}; V_C^{(\ell)}]$
 7:     Update concept outputs by one-directional attention:
 8:     $O_C^{(\ell)} = \mathrm{softmax}\Big(\frac{Q_C^{(\ell)}(\widehat{K}^{(\ell)})^\top}{\sqrt{d_h}}\Big)\widehat{V}^{(\ell)}$
 9:     Compute output-space saliency scores (per concept, per patch):
10:     $S_{k,i}^{(\ell)} = \langle O_{C,k}^{(\ell)}, O_{X,i}^{(\ell)}\rangle \quad \forall k \in [1..K],\ i \in [1..N]$
11:     Optionally normalize each $S_{k,:}^{(\ell)}$ with softmax over $i$ (patches)
12:     Retrieve / compute raw cross-attention map for each concept token if available (prompt-attention baseline)
13: end for
14: Average across layers: $\bar{S} = \frac{1}{|\mathcal{L}|}\sum_{\ell\in\mathcal{L}} S^{(\ell)}$
15: Reshape each $\bar{S}_{k,:}$ to $32\times 32$ to obtain $\mathrm{sal} \in \mathbb{R}^{K\times 32\times 32}$
16: Aggregate cached cross-attention similarly to obtain $\mathrm{cross} \in \mathbb{R}^{K\times 32\times 32}$
17: return $(\mathrm{sal}, \mathrm{cross})$
```

The relationship classes

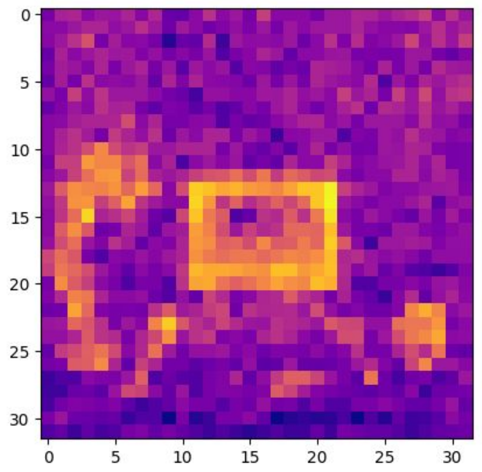
- Above
- around/near
- Behind
- Below
- Carrying
- Drinking
- Eating
- Hanging from
- Holding
- In
- In front of
- Left of
- Looking at
- on
- Playing with
- Pulling
- Pushing
- Riding
- Right of
- Sitting On
- Standing on
- Touching
- Using
- wearing

Sample maps [32x32]

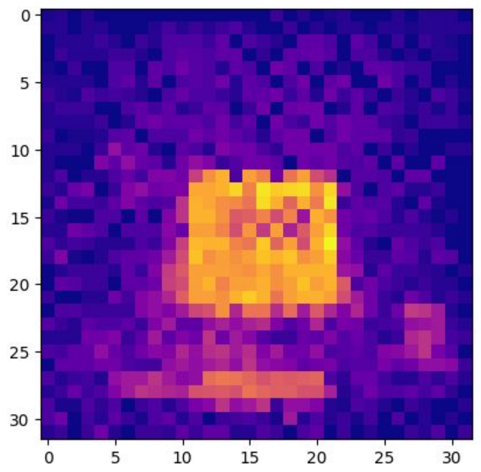
Prompt: a photo of laptop above desk on a plain background and **concept: laptop**



Early layers



Middle layers



Late layers

EXPERIMENTS

- With the dataset curated, we perform a surrogate classification task to **predict relationship class from the saliency and cross attention maps.**

- 3 classifiers:
  - low capacity -> MLP
  - medium capacity-> Tiny CNN
  - high capacity -> WideResnet [**SOTA on CIFAR]**

We hypothesize that, if our classifier is able to perform well using the saliency maps, the features captured by those layers is already strong with respect to the structural relationship of the prompt.

EXPERIMENTS

1. WideResNet (WRN-28-w)
A deep residual network optimized for width.
Initial Stage: Conv 3×3 (Input → 16 channels).
Core Structure: 3 Groups of NetworkBlocks (Strides: 1, 2, 2).
Widths: [16×w,32×w,64×w].
Depth: Each group contains N BasicBlocks (N=4 for depth 28).
BasicBlock Design: BN → ReLU → Conv 3×3 → BN → ReLU → Conv 3×3 (+ Residual Shortcut).
Classifier Head: BN → ReLU → AvgPool2d(8) → Linear (64×w → Classes).

2. TinyCNN
A compact convolutional network for efficient inference.
Layer 1: Conv 3×3 (C→32) + BN + ReLU.
Layer 2: Conv 3×3 (32→64) + BN + ReLU → MaxPool2d(2).
Layer 3: Conv 3×3 (64→128) + BN + ReLU → MaxPool2d(2).
Output: AdaptiveAvgPool2d(1) → Flatten → Linear (128→ Classes).

3. SimpleMLP
A standard Multi-Layer Perceptron baseline.
Input: Flatten (C×32×32 features).
Hidden Layer 1: Linear (→512) + ReLU + Dropout(0.3).
Hidden Layer 2: Linear (512→256) + ReLU + Dropout(0.3).
Output: Linear (256→ Classes).

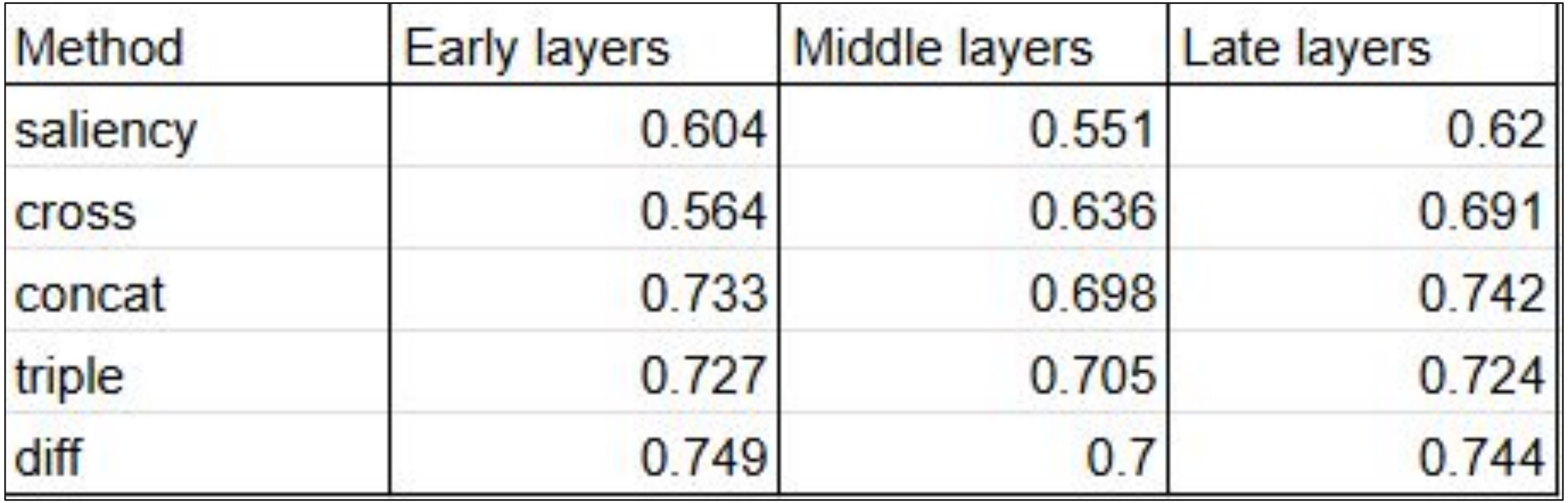
Fusion between saliency maps and cross-attention maps

Apart from individually checking performance on the maps, we also try to fuse and check the performance on fusion of the maps.

Different ways to fuse include:

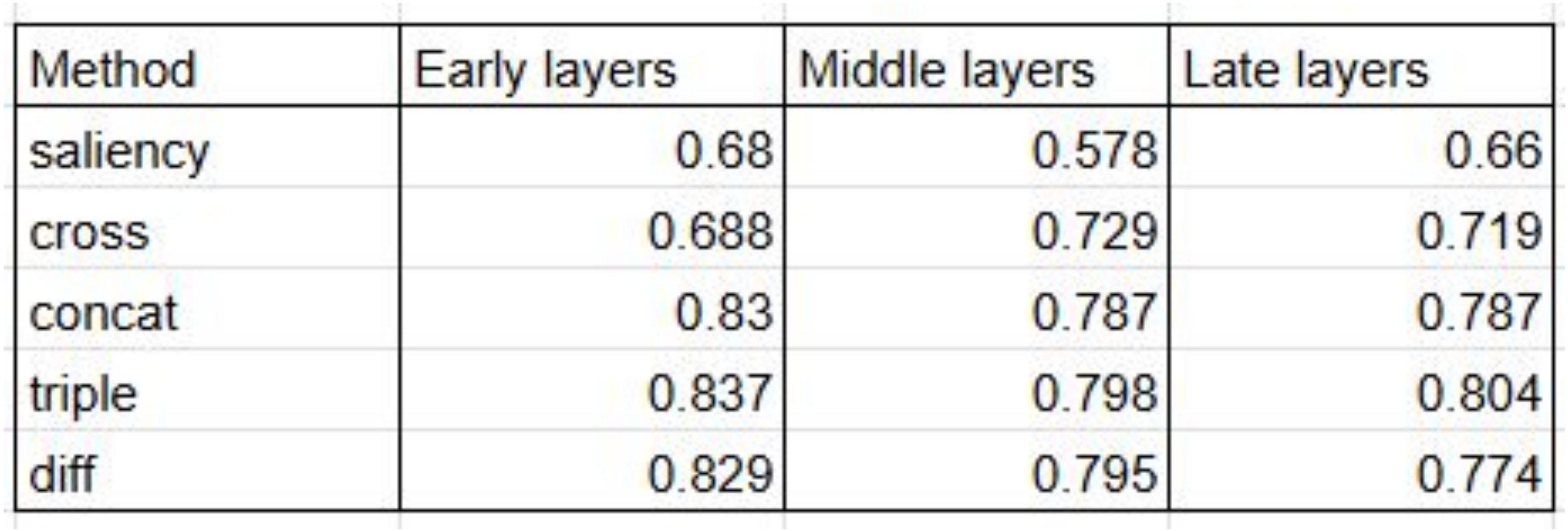
- **Concatenation:** simply concatenate the feature maps making it a 6 channel input to the classifier (instead of just 3)
- **Triple Fusion**: concatenate [saliency maps, cross-attention maps, saliency * cross-attention] (9 x 32 x 32)
- **Difference fusion:** concatenate [saliency maps, cross attention - saliency maps] (6 x 32 x 32)

Classifier ACCURACY - MLP



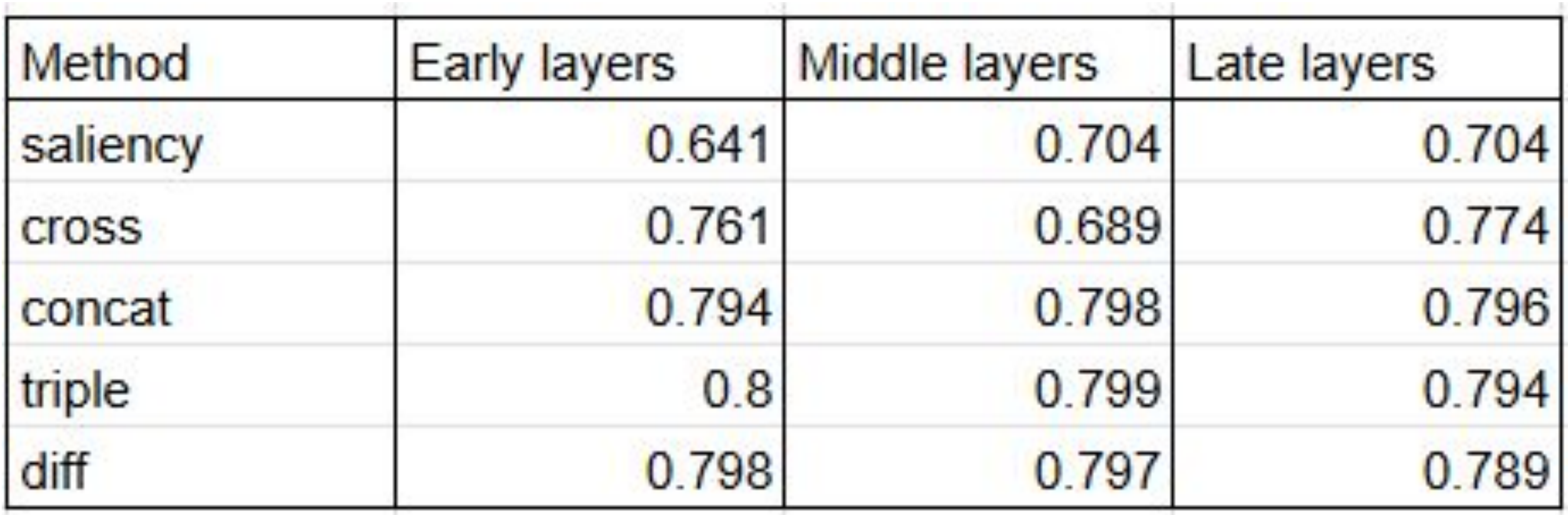
Method	Early layers	Middle layers	Late layers
saliency	0.604	0.551	0.62
cross	0.564	0.636	0.691
concat	0.733	0.698	0.742
triple	0.727	0.705	0.724
diff	0.749	0.7	0.744

Classifier ACCURACY - Tiny CNN



Method	Early layers	Middle layers	Late layers
saliency	0.68	0.578	0.66
cross	0.688	0.729	0.719
concat	0.83	0.787	0.787
triple	0.837	0.798	0.804
diff	0.829	0.795	0.774

Classifier ACCURACY - WideResnet



Method	Early layers	Middle layers	Late layers
saliency	0.641	0.704	0.704
cross	0.761	0.689	0.774
concat	0.794	0.798	0.796
triple	0.8	0.799	0.794
diff	0.798	0.797	0.789

ANALYSIS

Based on the different experimentations and results obtained, we have concluded the following with an explicit focus on saliency maps:

- **WideResnet model performs well on middle and later layers as the model capacity is good**
- **For low and medium capacity models, middle layers performs the worse while late layers and early layers perform comparably well.**
- **Different experiments on different model capacities and different concatenation techniques lead us to the following inference.**

INFERENCE

- We infer from the experiments that the middle layers attention maps do not carry enough information that can be exploited to predict the structure from them.

- We hypothesize this as an absence of “structural information” in the middle layers and try to introduce the graph embeddings in these layers using LoRA in upcoming experiments.

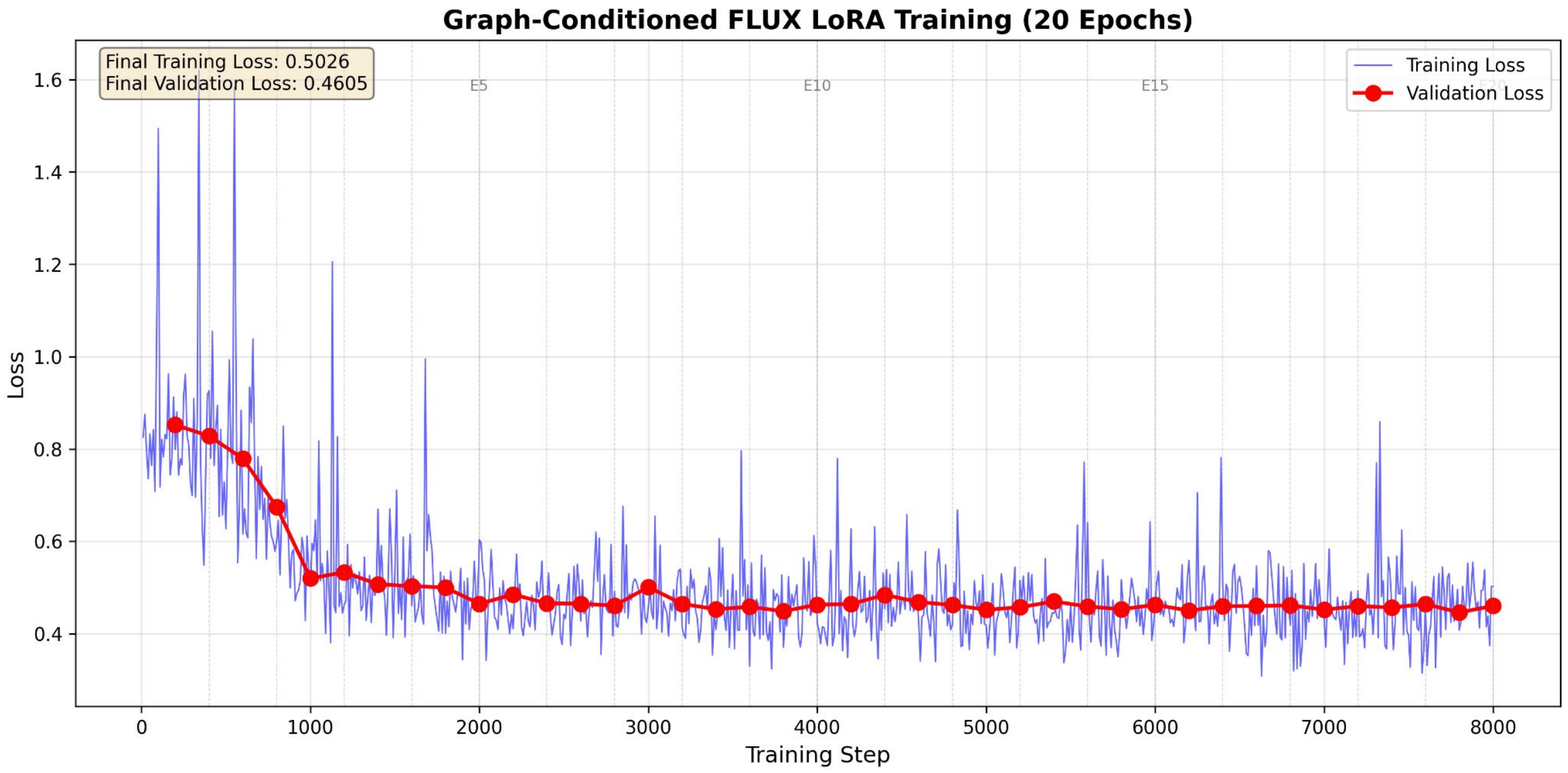
Graph-Conditioned LoRA for Relation Control in FLUX.1-schnell

Target: dual-stream (double) transformer blocks 7–12; relation supervision via ConceptAttention saliency + frozen 24-class classifier

Problem: middle blocks show weaker linear separability for relationship classification (vs early blocks)
Goal: condition FLUX.1-schnell generation on scene-graph triples (s,p,o) via graph tokens + LoRA adapters
Adaptation scope: LoRA on double blocks 7–12 (attention projections only); freeze base model, text encoders, VAE, graph encoder
Training outcome: generative loss converged, but relationship correctness did not improve under semantic evaluation

Training converged (Val loss 0.98→0.46), CLIP alignment 49.4% [poor].

Sources: diffusers FluxTransformer2DModel docs; ConceptAttention (arXiv:2502.04320); Visual Genome (arXiv:1602.07332); DreamRelation (ICCV 2025)


Graph-Conditioned FLUX LoRA Training (20 Epochs)
Final Training Loss: 0.5026
Final Validation Loss: 0.4605
Training Loss
Validation Loss
E5
E10
E15
E20
Loss
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
1000
2000
3000
4000
5000
6000
7000
8000
Training Step

Experimental Setup

- Dataset: Visual Genome triples (subject, predicate, object) with images
- Predicate canonicalized to 24 relation classes (geometric + semantic)
- Train/Val: 800 / 160 samples; balanced across 16 supported predicates (subset of 24 canonical)
- Resolution: 256x256
- Prompt template: "a photo of {subj} {pred_phrase} {obj} on a plain background"
- ConceptAttention produces saliency maps for [subject, predicate, object] tokens

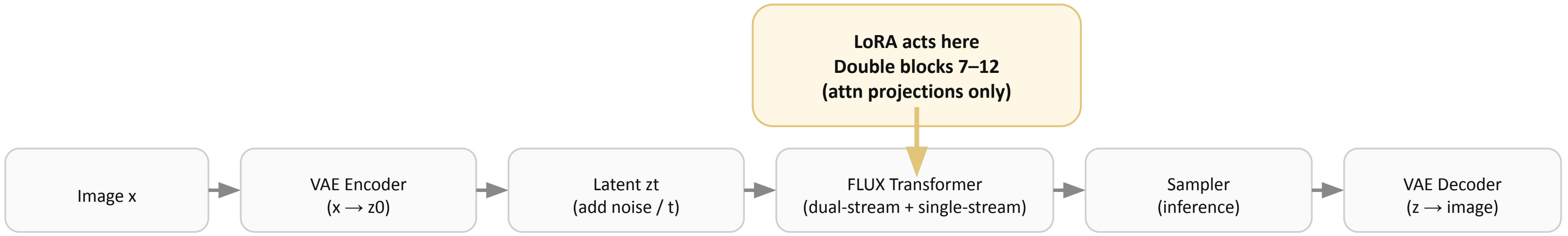
Graph conditioning module: SGDiff CGIP graph encoder → local tokens + global token; tokens are prepended to transformer context.

Hypothesis: middle blocks under-utilize explicit relation structure.

Design requirement:
Update a minimal parameter subset to improve relation handling without overwriting global generation behavior.

Sources: ConceptAttention (arXiv:2502.04320); Visual Genome (arXiv:1602.07332)

FLUX Latent-Space Architecture: Where LoRA Acts



Conditioning: prompt embeddings + graph tokens (local + global) prepended to context

Base model: FLUX.1-schnell rectified-flow DiT; dual-stream (19 double blocks) and 4-step inference

Training: compute losses in latent space using VAE-encoded ground-truth images (no multi-step sampling inside backprop)

Adaptation location: LoRA only in double blocks 7–12 (mid-to-late)

Evaluation: run full sampling and decode to pixels for qualitative panels

Sources: diffusers FluxTransformer2DModel docs (num_layers=19, num_single_layers=38)

Training Protocol: Teacher-Forced Updates; Full Sampling for Evaluation

Rationale: sampling is an inference-time integrator; training uses per-timestep regression in latent space.

Training (per step)

Encode image with VAE → latent $z0$ (cached)
Sample $t \sim U[0.5,1.0]$; construct $zt=(1-t)z0+t\epsilon$
Predict flow $v\theta(zt,c,g)$; optimize MSE flow loss only
Update: LoRA (blocks 7–12) + graph-token projector only

Evaluation (reporting)

Inference: 4 steps, no CFG, seeds {0,1,2}
Validation set: 160 triples × 3 seeds = 480 images
Metrics: CLIP alignment + margin; MSE sensitivity for $g+$ vs $g-$

Sources: diffusers FluxTransformer2DModel docs; FLUX.1-schnell model card; Visual Genome

LoRA Placement: Minimal, Targeted Adaptation

Target modules

- Apply LoRA only to attention projections in double blocks 7–12:
- Wq, Wk, Wv, Wo $\longrightarrow$ attn.to_q, attn.to_k, attn.to_v, attn.to_out.0
- Rank = 16, Alpha = 16
- Target blocks: double blocks 7–12
- Trainable params: ~10–12M (~0.1% of base)
- Frozen: base FLUX, text encoders, VAE, graph encoder

LoRA on a linear layer

```
W′ = W + (α/r)AB
A ∈ R^{d_in×r},  B ∈ R^{r×d_out},
r ≪ min(d_in,d_out)
```

Only A,B are trainable.

Attention rationale:
Q,K control routing (who attends to whom).
V control write-back content.
Relations primarily depend on token-to-token interaction.

Sources: diffusers FLUX LoRA guidance; DreamRelation (ICCV 2025) analysis of Q/K/V roles in MM-DiT

Optimization Objective

Total loss

$$L = L_{gen} + \lambda L_{rel\text{-}rank} + \alpha L_{gen\text{-}rel}$$

L_gen (generative anchor)

- Teacher-forced latent regression at sampled t
- Uses ground-truth image latents z0 (via VAE encoder)
- Prevents drift from the pretrained data manifold

L_rel-rank (relation contrastive)

- Frozen classifier on ConceptAttention saliency
- Same (zt,t,c) for g+ and g-
- Margin ranking encourages: score(g+) > score(g-)

$$L_{rel\text{-}rank} = \max(0, m - \ell^{+}_{y} + \ell^{-}_{y})$$

$$L_{gen\text{-}rel} = \| W \odot (\text{pred} - \text{target}) \|^{2}$$
$$W = \text{detach}(\text{normalize}(\sigma(S_{p}^{+})))$$

Detach(W) blocks trivial solutions via saliency manipulation.

Sources: ConceptAttention (arXiv:2502.04320) for saliency extraction; DreamRelation (ICCV 2025) for relation-centric contrastive loss design

Optimization Objective

Total loss

```
L = L_gen + λ L_rel-rank + α L_gen-rel
```

- **Disabled terms:** Lrel-rank and Lgen-rel (weights set to 0)
- **Reason (implementation decision):** contrastive supervision not active in the final run, proceeded with pure generative objective
- Frozen saliency-classifier trained on **fully denoised** generations
- during LoRA training saliency is extracted at **noisy intermediate zt** → strong distribution shift.
- Class logits were trained for **24-way predicate classification**, not calibrated as a **graph-match quality score** for ranking.

Sources: ConceptAttention (arXiv:2502.04320) for saliency extraction; DreamRelation (ICCV 2025) for relation-centric contrastive loss design

Contrastive Supervision via Negative Graphs

Used in evaluation; contrastive training term was disabled in final run.

Directional predicates

- Examples: left/right, above/below, in-front-of/behind
- Negative: swap subject/object
- $g^- = (o, p, s)$
- Loss penalizes supporting the original class y under swapped graph

Non-directional / confusable

- Examples: holding, wearing, looking-at, on, under, inside-of
- Negative: predicate replacement
- $g^- = (s, p', o)$
- Choose p′ as a hard negative (top confusions of frozen classifier)

Sources: Visual Genome (arXiv:1602.07332) for relationship annotations; DreamRelation (ICCV 2025) for relation-centric contrastive loss

Training Loop

Inputs (per sample): $z0 = \mathrm{VAEEnc}(x)$ (cached), prompt embedding (cached), graph tokens g

Timestep sampling: $t \sim U[0.5, 1.0]$

Noisy latent: $zt = (1-t) \cdot z0 + t \cdot \varepsilon$

Forward: predict flow/velocity $v\theta(zt, t, \text{prompt}, g)$

Loss used: $L = Lgen = \mathrm{MSE}(v\theta, v^*)$

Trainable: LoRA only (double blocks 7–12; attn.to_q/to_k/to_v/to_out.0) + graph-token projector

Disabled: λrel-rank = 0, αgen-rel = 0

Why disabled (observed):

- Classifier trained on saliency from fully denoised generations; during LoRA training saliency is extracted from noisy intermediate zt (distribution shift).
- Classifier logits are optimized for 24-way predicate ID; not calibrated as a graph-match quality score at intermediate noise levels.

Executed objective: generative flow MSE only; preference terms disabled due to ranking instability.

Configuration and Compute

Dataset: Visual Genome; Train 800 / Val 160; 256×256; 16 supported predicates

Optimization: AdamW, lr=1e−4, wd=1e−2, grad clip=1.0, bf16

Training: 20 epochs, batch size 2; 400 steps/epoch; 8000 total steps

Runtime: ~17 min/epoch on A6000; ~5.7 hours total

LoRA: rank 16, alpha 16; target double blocks 7–12; attention projections only

See LORA_EXPERIMENT_REPORT.md for exact hyperparameters and timings.

Evaluation Protocol

Inference: 4 steps (schnell), guidance scale 0.0, seeds {0,1,2}

Samples: 160 val triples × 3 seeds = 480 images

For each triple (s,p,o):

- g+ : ground-truth relation
- g− : corrupted relation
  - Directional swap: “dog riding horse” → “horse riding dog”
  - Predicate replacement: “dog holding ball” → “dog wearing ball”
- Generate *I+* = sample(g+), *I−* = sample(g−) with the same seed

Metrics: CLIP alignment (pos vs neg), CLIP margin; MSE sensitivity $||(I+) - (I-)||^2$

Evaluation uses full sampling; negatives constructed by swap and predicate replacement.

Results

Training convergence (val loss): 0.98 → 0.46 (epoch 1 → 20)

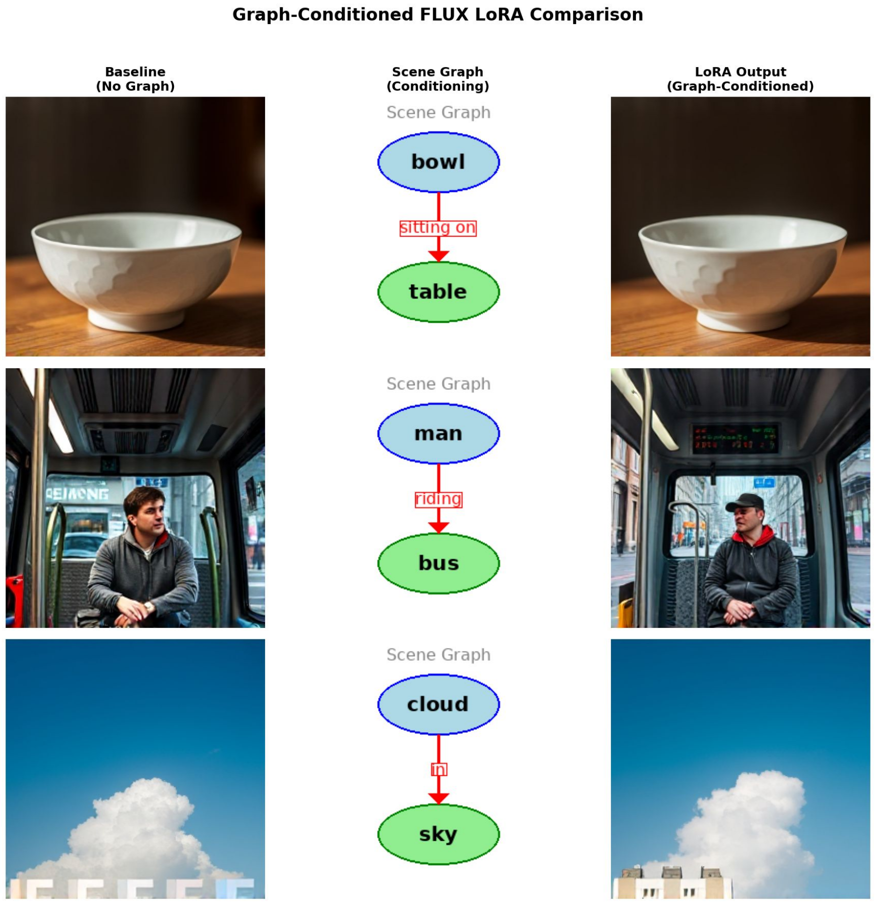
Graph sensitivity: MSE(g+, g−) mean 8613.7, std 5267.9

Semantic relationship correctness (CLIP):

- Accuracy 49.4% (≈ random)
- Margin (pos − neg): −0.022 (negatives sometimes score higher)

Interpretation: outputs change with graph perturbations, but preference for the correct relation is not recovered under semantic scoring.

Quantitative values reported on 160-val × 3 seeds (480 images).


Graph-Conditioned FLUX LoRA Comparison
Baseline
(No Graph)
Scene Graph
(Conditioning)
LoRA Output
(Graph-Conditioned)
Scene Graph
bowl
sitting on
table
Scene Graph
man
riding
bus
Scene Graph
cloud
in
sky

Failure Analysis and Limitations

Observed:

Training converged but semantic evaluation did not improve (CLIP ≈ 49.4%).

Model is graph-sensitive (high MSE between g+ and g−) but not relation-correct.

Primary causes:

No preference supervision: Lgen does not enforce g+ ≻ g−.

Contrastive term (Lrel-rank) disabled; no alternative preference loss active.

Graph encoder frozen; limited ability to refine relation embeddings.

Prompt contains relation text, reducing reliance on graph tokens.

Spatial evaluation via detection+bboxes was blocked by predicate/vocab limitations.

Next iteration:

Use neutral prompts (“dog and horse”) to isolate graph effect.

Re-introduce a preference loss calibrated for intermediate zt, or move ranking to final sampled images with an efficient estimator.

Limitations and next steps summarized from the experiment report.

Ablation: 10 vs 20 Epochs

10 epochs (10 samples): MSE 6460; CLIP 60% (weak); margin 1.01

20 epochs (160 samples): MSE 8613; CLIP 49%; margin −0.022

Conclusion: longer training increased sensitivity but degraded semantic alignment without preference supervision.

Ablation values from the experiment report.

THANK YOU